AF

TRANSMITTAL OF APPEAL BRIEF (Large Entity)	Docket No. 9491(NCR.0031US)

In Re Application Of: **Ramesh Bhashyam et al.**

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
09/851,592	05-09-2001	Diem K. Cao	26890	2194	2588

Invention: **Event-Based Synchronization**

COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on **October 5, 2005.**

The fee for filing this Appeal Brief is: **$500.00**

☐ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **14-0225 (9491)**

☐ Payment by credit card. Form PTO-2038 is attached.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Signature

Dated: **November 29, 2005**

Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Suite 100
Houston, TX 77024
Telephone: (713) 468-8880, ext. 304
Facsimile: (713) 468-8883

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on

11-29-2005
(Date)

Signature of Person Mailing Correspondence

Ginger Yount

Typed or Printed Name of Person Mailing Correspondence

cc:

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Ramesh Bhashyam et al.	§	Art Unit:	2194
Serial No.:	09/851,592	§		
		§	Examiner:	Diem K. Cao
Filed:	May 9, 2001	§		
For:	Event-Based Synchronization	§	Atty. Dkt. No.:	9491 (NCR.0031US)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF PURSUANT TO 37 C.F.R § 41.37

Sir:

The final rejection of claims 2-6, 9, 12-26, and 37-42 is hereby appealed.

I. REAL PARTY IN INTEREST

The real party in interest is the NCR Corporation.

II. RELATED APPEALS AND INTERFERENCES

None.

Date of Deposit: November 29, 2005

I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313.

Ginger Yount

III. STATUS OF THE CLAIMS

Claims 2-6, 9, 12-26, and 37-42 have been finally rejected and are the subject of this appeal.

Claims 1, 7, 8, 10, 11, 27-36, 43, and 44 have been cancelled.

IV. STATUS OF AMENDMENTS

The Amendment after final submitted on July 26, 2005 will be entered for purposes of appeal. *See* 8/11/2005 Advisory Action.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the independent claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

Independent claim 9 recites a system comprising:

> a Unix operating system (Fig. 1:12; Fig. 2:104; Specification, p. 3, lines 3-5; p. 6, lines 6-7);
>
> a plurality of execution entities (Fig. 1:T1, T2, T3, T4, T5), the plurality of execution entities including a first execution entity (Specification, p. 3, lines 1-23);
>
> an event control module (Fig. 1:28; Fig. 2:120) adapted to create event objects (Fig. 3:200; Fig. 6:500, 502) representing respective events each having a state, the first execution entity to wait on plural events (Specification, p. 4, lines 3-19; p. 6, lines 19-26);

a data structure (Fig. 6:504) associated with the first execution entity, the data structure containing information of the plural events that the first execution entity is waiting on, the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events (Specification, p. 8, lines 10-24); and

a controller (Fig. 3:201) adapted to awaken the first execution entity by signaling the first execution entity in response to one or more event state changes of the states of the plural events according to the logical relationship specified by the indicator (Specification, p. 7, lines 16-19; p. 9, lines 3-11),

wherein each event object contains an indication (Fig. 3:202) of the state of the event (Specification, p. 6, lines 27-30),

wherein the indication has a first state to indicate that the event has been signaled and a second state to indicate that the event has not been signaled (Specification, p. 5, lines 11-13),

wherein each event object has a type indication (Fig. 3:204) to indicate whether the event object state indication is to be automatically reset to the second state from the first state once the event has been signaled or to be manually reset to the second state from the first state by an explicit action (Specification, p. 5, lines 13-19; p. 7, lines 21-30).

Independent claim 21 recites an article comprising at least one storage medium containing instructions for providing event-based synchronization in a system in which execution entities (Fig. 1:T1, T2, T3, T4, T5) are running, the instructions when executed causing the system to:

> generate event objects (Fig. 3:200; Fig. 6:500, 502) in a Unix operating system (Fig. 1:12; Fig. 2:104) environment representing events used for synchronizing execution entities in the system, each event object having a state (Fig. 3:202) to indicate if the corresponding event has been signaled (Specification, p. 5, lines 11-13; p. 6, lines 27-30);
>
> provide a queue (Fig. 3:205) containing entries associated with a first event object, each entry associated with a corresponding execution entity, the plural entries of the queue enabling plural execution entities to wait on the first event object (Specification, p. 6, line 30-p. 7, line 8); and
>
> selectively set a type variable (Fig. 3:204) to one of a first value and a second value, the first value indicating that the first event object is of an auto-reset type, and the second value indicating that the first event object is of a manual reset type (Specification, p. 5, lines 13-19; p. 7, lines 21-30);
>
> in response to the state of the first event object indicating the corresponding event has been signaled,
>
> > automatically clear the state of the first event object to an un-signaled state and awaken only one of the plural execution entities waiting on the first event object in response to the type variable being set to the first value (Specification, p. 7, lines 23-26), and
> >
> > not clear the state of the first event object until manually cleared and awaken all threads waiting on the first event object in response to the type variable being set to the second value (Specification, p. 7, lines 26-30).

Independent claim 37 recites a system comprising:

> a Unix operating system (Fig. 1:12; Fig. 2:104; Specification, p. 3, lines 3-5; p. 6, lines 6-7);
>
> a plurality of execution entities (Fig. 1:T1, T2, T3, T4, T5, Specification, p. 3, lines 1-23);

a storage module (Fig. 1:26; Fig. 2:106) containing an event library (Fig. 1:28; Fig. 2:104; Specification, p. 4, lines 3-30); and

a processor (Fig. 1:18; Fig. 2:116) adapted to execute the event library to provide an event-based synchronization mechanism comprising one or more events on which the plural execution entities are able to sleep (Specification, p. 3, line 24-p. 4, line 8).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claim 40 Was Rejected Under 35 U.S.C. § 112, ¶ 2.

B. Claims 2-6, 9, 12-26, And 39-42 Were Rejected Under § 103 Over U.S. Patent No. 6,128,640 (Kleinman) In View Of U.S. Patent No. 5,598,562 (Cutler) And U.S. Patent No. 6,546,443 (Kakivaya).

C. Claims 37 And 38 Were Rejected Under § 103 Over Kleinman Alone.

VII. ARGUMENT

The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.R. § 41.37(c)(1)(vii).

A. Claim 40 Was Rejected Under 35 U.S.C. § 112, ¶ 2

1. Claim 40.

Claim 40 was rejected for being indefinite. However, in the Amendment after final, claim 40 was amended to address this rejection. Since the Amendment has been entered for purposes of appeal, it is respectfully submitted that the § 112 rejection has been overcome.

Reversal of the final rejection of the above claim is respectfully requested.

B. Claims 2-6, 9, 12-26, And 39-42 Were Rejected Under § 103 Over U.S. Patent No. 6,128,640 (Kleinman) In View Of U.S. Patent No. 5,598,562 (Cutler) And U.S. Patent No. 6,546,443 (Kakivaya).

1. Claims 21-26.

It is respectfully submitted that a *prima facie* case of obviousness has not been established with respect to claim 21 for at least the reason that no motivation or suggestion existed to combine the teachings of Kleinman, Cutler, and Kakivaya. *See* M.P.E.P. § 2143 (8th ed., Rev. 3), at 2100-135.

Claim 21 recites generating event objects in a Unix operating system environment and the provision of a queue containing entries associated with a first event object, each entry associated with a corresponding execution entity, where the plural entries of the queue enable plural execution entities to wait on the first event object. Moreover, claim 21 recites that a type variable is selectively set to one of a first value and a second value, with the first value indicating that the first event object is of an auto-reset type, and the second value indicating that the second event object is of a manual reset type. In response to the state of the first event object indicating the corresponding event has been signaled, the system automatically clears the state of the first event object to an un-signaled state and awakens only one of the plural execution entities waiting on the first event object in response to the type variable being set to the first value. However, if the type variable is set to the second value, then the state of the first event object is not cleared until manually cleared, and all threads waiting on the first event object are awakened.

This type variable is not disclosed or suggested by either Kleinman or Cutler, a point that was conceded by the Examiner. *See* 5/26/2005 Office Action at 14-15. The Examiner relied upon Kakivaya as teaching such a type variable. *Id.*

Appellant respectfully submits that there existed no motivation to combine the teachings of Kleinman, Cutler, and Kakivaya. In the Advisory Action dated August 11, 2005, the Examiner cited two cases as purportedly supporting the Examiner's obviousness rejections: *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988), and *In re Jones*, 958 F.2d 347, 21 U.S.P.Q.2d 1941 (Fed. Cir. 1992). These cases clearly do not support the Examiner's obviousness rejections. *In re Fine* held that obviousness "cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination." *In re Fine*, 837 F.2d at 1074. *In re Jones* held that "[b]efore the PTO may combine the disclosures of two or more prior art references in order to establish *prima facie* obviousness, there must be some suggestion for doing so, found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art." *In re Jones*, 958 F.2d at 351.

There are at least two reasons that a person of ordinary skill in the art would not have been motivated to combine the teachings of Kleinman, Cutler, and Kakivaya. First, Kleinman relates to the Unix operating system, whereas Kakivaya relates to a Windows operating system. Second, Kleinman uses a notify_all function to unblock (or awaken) all threads waiting for a particular event object. Kleinman, 5:53-54, 8:37-41. Kleinman makes no suggestion whatsoever of any need to selectively awaken just one thread, or awaken all threads, based on the type of event object. None of the cited references, Kleinman, Cutler, or Kakivaya, suggest any desirability to modify Kleinman to achieve the claimed invention.

It is well established law that "[t]he mere fact that the prior art could be so modified would not have made the modification **obvious** unless the prior art suggested the **desirability** of the modification." *In re Gordon,* 733 F.2d 900, 902, 221 U.S.P.Q. 1125 (Fed. Cir. 1984)

(emphasis added). As the Federal Circuit has stated, "virtually all [inventions] are combinations of old elements." *In re Rouffet*, 149 F.3d 1350, 1357, 47 U.S.P.Q.2d 1453 (Fed. Cir. 1998). "Most, if not all, inventions are combinations and mostly of old elements." *Id.*

> Therefore an examiner may often find every element of a claimed invention in the prior art. If identification of each claimed element in the prior art were sufficient to negate patentability, very few patents would ever issue. Furthermore, rejecting patents solely by finding prior art corollaries for the claimed elements would permit an examiner to use the claimed invention itself as a blueprint for piecing together elements in the prior art to defeat the patentability of the claimed invention. Such an approach would be 'an illogical and inappropriate process by which to determine patentability.'

Id.

The only basis for the proposed combination of reference teachings made by the Examiner is the disclosure of the present invention. However, relying upon the present invention to piece together un-related elements of prior art references constitutes impermissible hindsight. As warned by the *In re Fine* case cited by the Examiner, "[o]ne cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention." *In re Fine*, 837 F.2d at 1075. Kleinman is unambiguous in indicating the use of the notify_all() routine associated with an event object to unblock *all* threads waiting for the event. Kleinman, 8:37-41. There is no indication provided anywhere in Kleinman of any need or desirability to employ a different routine for the event object that can selectively notify only one thread or all threads based on a type variable. If the benefits of modifying Kleinman as proposed by the Examiner were so apparent, as the Examiner contended, then Kleinman would have proposed the use of a different routine associated with an event object that enables selective notification of only one thread or of all threads upon signaling of the event. The fact that Kleinman does not even hint at using such a different routine is strong objective evidence that a

person of ordinary skill in the art would not have been motivated to modify Kleinman to incorporate the teachings of Kakivaya.

The Examiner further argued that Kakivaya teaches that various substitutes to the Microsoft Windows operating system could be used in Kakivaya's system. 5/26/2005 Office Action at 19. The Examiner further argued that Kakivaya's "invention is not limited to only Window operating system (based on the claim and abstract of the patent)." *Id.* This was a basis for the assertion by the Examiner that "one of ordinary skill in the art could apply the technique and modify it to work in different type of operating system." *Id.*

Appellant respectfully disagrees. Note that the abstract and claims of Kakivaya do not add teachings in addition to the teachings of the specification of Kakivaya, which is focused on the Windows operating system. Also note that the Kakivaya patent is owned by Microsoft Corporation, which makes Windows operating systems. Thus, clearly, the focus of the Microsoft patent is the Windows operating system. A person of ordinary skill in the art would not have been motivated to apply techniques of the Windows operating system to the Unix operating system environment described in Kleinman.

Cutler does not provide any other suggestion or teaching that would have motivated a person to apply the teachings of Kakivaya to Kleinman.

In the Advisory Action dated August 11, 2005, the Examiner further stated that "all references relate to event synchronization in the operating system," and therefore "it would have been obvious to combine the references to improve the system of Kleinman to be more flexible." 8/11/2005 Advisory Action at 2. This does not change the fact that there still did not exist any suggestion of any desirability to incorporate the teachings relating to automatic and manual events disclosed in Kakivaya into the system of Kleinman. As noted above, Kleinman teaches

specifically the use of a notify_all() routine associated with an event to unblock all threads waiting for the event, with no indication provided anywhere of any desirability to incorporate a feature in which an event object can selectively notify only one thread or all threads based on a type variable. The suggestion to incorporate the type variable of Kakivaya into the Kleinman/Cutler system is found only in the teachings of the present invention. Without the present invention, a person of ordinary skill in the art would not have been motivated to combine the teachings of Kleinman, Cutler, and Kakivaya. In fact, the three references cited by the Examine constitute objective proof that persons of ordinary skill in the art at the time of the present invention were not motivated to combine the individual elements taught by the three references into the combination recited in claim 21. Despite the teachings of the individual elements provided by Kleinman, Cutler, and Kakivaya, none of the references provided any suggestion to combine these individual elements. In view of the foregoing, it is respectfully submitted that claim 21 is not obvious over the references.

Therefore, withdrawal of the final rejection of the above claims is respectfully requested.

2. Claims 2-6, 9, 12-20, and 39-42.

Independent claim 9 was also rejected as being obvious over the asserted combination of Kleinman, Cutler, and Kakivaya. Claim 9 recites a system that comprises a Unix operating system, a plurality of execution entities that include a first execution entity, and an event control module to create event objects representing respective events each having a state, the first execution entity to wait on plural events, and a data structure associated with the first execution entity, where the data structure contains information of the plural events that the first execution entity is waiting on, the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events. Also, claim 9

recites that each event object has a type indication to indicate whether the event object state indication is to be automatically reset to the second state from the first state once the event has been signaled or to be manually reset to the second state from the first state by an explicit action.

As conceded by the Examiner, Kleinman does not teach an event object having the recited type indication. 5/26/2005 Office Action at 13. However, as with claim 21, the Examiner relied upon Kakivaya as teaching the type indication. *Id.*

For the reasons stated above with respect to claim 21, it is respectfully submitted that no motivation or suggestion existed to combine the teachings of Kleinman, Cutler, and Kakivaya. The *prima facie* case of obviousness of claim 9 is defective for at least this reason.

Moreover, the Examiner also conceded that Kleinman fails to disclose a data structure containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events, where a controller is adapted to awaken the first execution entity by signaling the first execution entity in response to one or more event state changes of the states of the plural events according to the logical relationships specified by the indicator. 5/26/2005 Office Action at 3-4. However, the Examiner cited Cutler as disclosing such a feature. *Id.* at 4.

It is respectfully submitted that there did not exist any motivation to incorporate the teachings of Cutler with respect to the Executive Wait Multiple routine that has two modes of operation (disclosed in column 28 of Cutler) into the Kleinman system. The Executive Wait Multiple routine of Cutler can either (1) suspend the issuing program until any of the specified waitable objects become signaled, or (2) suspend the issuing program until all of the specified waitable objects become signaled. Cutler, 28:24-29. In contrast, Kleinman states that "the thread effectively waits on the logical inclusive OR of the events represented by all of the Alert

objects in the container." Kleinman, 4:32-34. Also, Kleinman states that "the thread, in effect, blocks until *any* of the Alerts in the container collection has been notified." Kleinman, 5:28-31 (emphasis added). Kleinman further teaches that "when a thread has collected Alerts into a container Alert, it can wait for *any* one of them with a wait_for_any() routine of that container Alert." Kleinman, 6:37-39 (emphasis added). Thus, it is clear from these various passages of Kleinman that Kleinman is directed to a thread waiting on one specific logical relationship of multiple events: a logical OR relationship. There is no teaching or suggestion of any desirability of a different logical relationship made by Kleinman. Despite the lack of any suggestion of the desirability to use a different logical relationship among events for a particular thread, the Examiner nevertheless stated that it would have been obvious to modify Kleinman with the teachings of Cutler. Again, this assertion impermissibly benefits from the teachings of the present invention. Without the teachings of the present invention, a person of ordinary skill in the art would not have been motivated to modify Kleinman with the teachings of Cutler. Therefore, this a further reason that claim 9 is not obvious over Kleinman, Cutler, and Kakivaya.

In view of the foregoing, the final rejection of the above claims is respectfully requested.

3. Claims 37 and 38.

Independent claim 37 was rejected as being obvious over Kleinman alone. The Examiner conceded that Kleinman fails to teach or suggest an event library to provide an event-based synchronization mechanism as recited in claim 37. 5/26/2005 Office Action at 18. Nevertheless, the Examiner stated that "[b]ecause there are multiple event classes in the system, it would have been obvious to one of ordinary skill in the art to put all the event classes in the library for easier maintenance and use." *Id.* However, the Examiner has failed to provide support in the form of objective evidence for the assertion that a person of ordinary skill in the art would have been

motivated to modify Kleinman to incorporate the event library for providing an event-based synchronization mechanism.

Despite Appellant's request for a reference that provided a suggestion to modify Kleinman to incorporate the recited event library, the Examiner failed to do so. Instead, the Examiner made the following statement:

> [A]lthough Kleinman does not teach a library, but a library is a collection of files/routines that can be used by any program, since the system of Kleinman can be used by any process/application, it would have been obvious to one of ordinary skill in the art to put all the classes in one place, i.e., library, for use by multiple applications, for easier maintenance.

8/11/2005 Advisory Action at 3.

Again, such a statement is conclusory in nature. Without objective evidence that a person of ordinary skill in the art would have been motivated to modify Kleinman to incorporate an event library that can be executed by a processor to provide an event-based synchronization mechanism that comprises one or more events on which plural execution entities are able to sleep, it is respectfully submitted that a prima facie case of obviousness has not been established with respect to claim 37.

In view of the foregoing, reversal of the final rejection of the above claim is respectfully requested.

VIII. CONCLUSION

In view of the foregoing, reversal of all final rejections and allowance of all pending claims is respectfully requested.

Respectfully submitted,

Date: Nov. 29, 2005

Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Suite 100
Houston, TX 77024
Telephone: (713) 468-8880
Facsimile: (713) 468-8883

APPENDIX OF APPEALED CLAIMS

The claims on appeal are:

2. The system of claim 9, wherein the event control module is adapted to define a queue for a first one of the event objects, the queue having plural entries corresponding to plural execution entities waiting on the event represented by the first event object.

3. The system of claim 2, wherein the event control module is adapted to further create second objects, wherein each entry of the queue comprises a link to a corresponding second object, each execution entity to sleep on an associated second object to wait on the event represented by the first event object.

4. The system of claim 3, wherein each second object is defined by a condition variable.

5. The system of claim 4, wherein the controller signals each thread by signaling the condition variable.

6. The system of claim 3, wherein each second object is defined by a condition variable and a mutex.

9. A system comprising:

a Unix operating system;

a plurality of execution entities, the plurality of execution entities including a first execution entity;

an event control module adapted to create event objects representing respective events each having a state, the first execution entity to wait on plural events;

a data structure associated with the first execution entity, the data structure containing information of the plural events that the first execution entity is waiting on, the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events; and

a controller adapted to awaken the first execution entity by signaling the first execution entity in response to one or more event state changes of the states of the plural events according to the logical relationship specified by the indicator,

wherein each event object contains an indication of the state of the event,

wherein the indication has a first state to indicate that the event has been signaled and a second state to indicate that the event has not been signaled,

wherein each event object has a type indication to indicate whether the event object state indication is to be automatically reset to the second state from the first state once the event has been signaled or to be manually reset to the second state from the first state by an explicit action.

12. The system of claim 9, further comprising queues associated with corresponding event objects representing events the first execution entity is waiting on, each queue containing an entry corresponding to the first execution entity.

13. The system of claim 12, wherein the event control module is adapted to define a barrier object, the first execution entity to sleep on the barrier object to wait on the plural events, the queue of each event object containing a link to the barrier object.

14. The system of claim 13, wherein the barrier object is defined at least by a condition variable.

15. The system of claim 13, wherein the barrier object is defined at least by a condition variable and a mutex.

16. The system of claim 9, wherein the event control module comprises a library.

17. The system of claim 9, wherein the execution entities comprise threads.

18. The system of claim 17, further comprising plural processes, each process associated with one or more threads,

the event control module to create a local event to synchronize threads within a process and to create a global event to synchronize threads of different processes.

19. The system of claim 18, wherein the global event comprises a named event.

20. The system of claim 9, further comprising a plurality of nodes, each node comprising one or more of the plurality of execution entities.

21. An article comprising at least one storage medium containing instructions for providing event-based synchronization in a system in which execution entities are running, the instructions when executed causing the system to:

generate event objects in a Unix operating system environment representing events used for synchronizing execution entities in the system, each event object having a state to indicate if the corresponding event has been signaled;

provide a queue containing entries associated with a first event object, each entry associated with a corresponding execution entity, the plural entries of the queue enabling plural execution entities to wait on the first event object; and

selectively set a type variable to one of a first value and a second value, the first value indicating that the first event object is of an auto-reset type, and the second value indicating that the first event object is of a manual reset type;

in response to the state of the first event object indicating the corresponding event has been signaled,

automatically clear the state of the first event object to an un-signaled state and awaken only one of the plural execution entities waiting on the first event object in response to the type variable being set to the first value, and

not clear the state of the first event object until manually cleared and awaken all threads waiting on the first event object in response to the type variable being set to the second value.

22. The article of claim 21, wherein the instructions when executed cause the system to further create barrier objects, each execution entity waiting on a corresponding barrier object to wait on an event.

23. The article of claim 22, wherein the instructions when executed cause the system to create barrier objects by defining each barrier object based on a condition variable according to the Unix operating system.

24. The article of claim 22, wherein the instructions when executed cause the system to create barrier objects by defining each barrier object based on a condition variable and mutex according to the Unix operating system.

25. The article of claim 22, wherein the queue of the first event object contains entries pointing to the barrier objects of the plural execution entities waiting on the first event object.

26. The article of claim 25, wherein the instructions when executed cause the system to provide a routine associated with each event object, the routine of the first event object to traverse the queue of the first event object and to signal the barrier objects pointed to by the entries in the queue of the first event object.

37. A system comprising:

a Unix operating system;

a plurality of execution entities;

a storage module containing an event library; and

a processor adapted to execute the event library to provide an event-based synchronization mechanism comprising one or more events on which the plural execution entities are able to sleep.

38. The system of claim 37, further comprising plural processes, each process associated with one or more of the execution entities, wherein the synchronization mechanism comprises a local event synchronization mechanism to synchronize execution entities associated with one process, and a global event synchronization mechanism to synchronize execution entities associated with plural processes.

39. The system of claim 9, wherein the indicator is settable to a first value to specify a logical AND relationship between the plural events, and in response to the first value of the indicator, the controller to awaken the first execution entity in response to all of the plural events waited on by the first execution entity being signaled.

40. The system of claim 39, wherein the indicator is settable to a second value to specify a logical OR relationship between the plural events, and in response to the second value of the indicator, the controller to awaken the first execution entity in response to any of the plural events waited on by the first execution entity being signaled.

41. The article of claim 21, wherein a first one of the execution entities waits on plural events represented by respective event objects, the instructions when executed causing the system to:

provide a data structure containing information of the plural events waited upon by the first execution entity, the data structure further containing an indicator settable to one of plural values to specify respective plural logical relationships between the plural events waited on by the first execution entity; and

awaken the first execution entity in response to states of the plural events waited upon by the first execution entity according to the logical relationship specified by the indicator.

42. The article of claim 41, wherein the instructions when executed cause the system to set the indicator to a value to indicate a logical AND relationship, wherein awakening the first execution entity is in response to all of the plural events waited upon by the first execution entity being signaled.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.